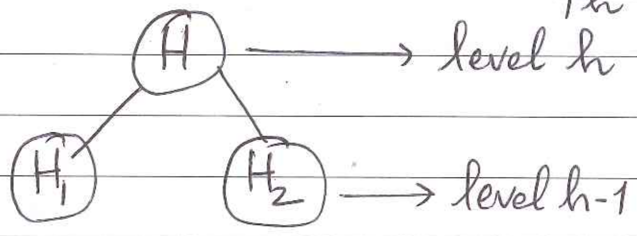
Lecture 6

Recall the Karger-Stein algorithm. We wanted to estimate the probability that Randomized-mincut (H, r) returns a min-cut in H under the assumption that our min-cut C is preserved in H.

We used p_h to denote the above probability.



We had the following recurrence relation:

$$p_h - p_{h-1} \geq -\frac{p_{h-1}^2}{4} \quad \text{for } h \geq 1$$

$$p_0 = 1.$$

Observe that $p_h = \frac{4}{h+4}$ almost fits the above recurrence.

That is, $p_h - p_{h-1}$ becomes $\frac{-4}{(h+4)(h+3)}$

and $-\frac{p_{h-1}^2}{4} = -\frac{4}{(h+3)(h+3)}$

and so we have $p_h - p_{h-1} \simeq -\frac{p_{h-1}^2}{4}$.

Hence $p_h \geq \frac{c_0}{h}$ for some constant c_0.

For the starting graph G, we have $h = 2\log n$. Moreover, the assumption that our min-cut C is preserved in G is obviously true since no edge contractions have happened in G, which is the given graph.

Thus $p_{2\log n} \geq \frac{c_0}{2\log n}$. Hence success probability of Karger-Stein algorithm is at least $c_0/2\log n$.

Repeat this algorithm $\frac{4\log n}{c_0}$ times and return the least size candidate cut.

Error probability $\leq \left(1 - \frac{c_0}{2\log n}\right)^{\frac{4\log n}{c_0}} \leq 1 - \frac{1}{e^2} \leq \frac{1}{4}$.

The total running time is $O(n^2 \log n \cdot \log n)$
$= O(n^2 \log^2 n)$.

Date

We will start a new topic now.

Max flow algorithms

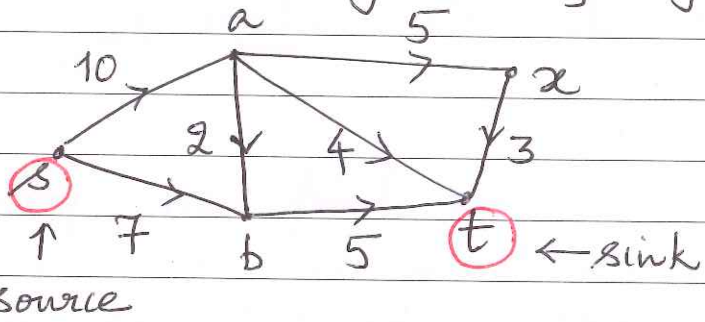
Input: a directed graph $G = (V, E)$ with m edges and n vertices

There are 2 special vertices s and t.

- s is called the source
- t is called the sink.

Every edge e has a non-negative capacity $c(e)$

An example:

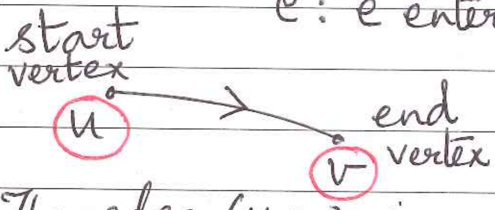


The numbers 10, 7, 2, ... are edge capacities.

Flow is a function $f : E \rightarrow \mathbb{R}_{\geq 0}$ that satisfies:

(1) $0 \leq f(e) \leq c(e) \quad \forall e \in E$

(2) $\displaystyle\sum_{e:\, e \text{ entering } u} f(e) = \sum_{e:\, e \text{ leaving } u} f(e) \quad \forall u \in V - \{s, t\}$.



The edge (u, v) is leaving u and entering v.

Goal: Maximize the net flow into t

That is, maximize the quantity

$$\boxed{\sum_{e:\, e \text{ entering } t} f(e) - \sum_{e:\, e \text{ leaving } t} f(e)}$$

In our example above, such a flow f is:

$f((s, b)) = f((b, t)) = 5$, ~~$f((s, a))$~~ $f((a, x)) = f((x, t)) = 3$,

$f((s, a)) = 7$, $f((a, t)) = 4$, $f(e) = 0$ for other edges e

So the max flow value in our example is $5 + 3 + 4 = 12$.

Date

<u>Exercise</u>. For any vertex u in G, define

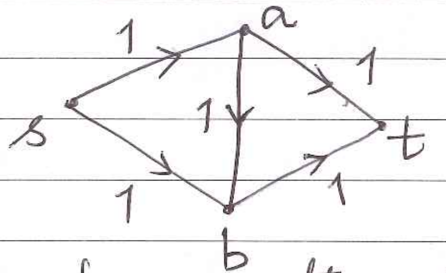
$$\text{excess}(u) = \sum_{e:\, e \text{ entering } u} f(e) - \sum_{e:\, e \text{ leaving } u} f(e)$$

Show that $\text{excess}(s) = -\,\text{excess}(t)$ where s is the source and t is the sink.

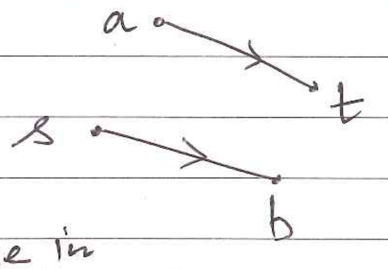
Our goal is to compute a flow function f that maximizes excess(t). How do we find such a function f?

A first idea. Say, we find an s-t path in G and send as much flow along it as possible.
- Then we delete saturated edges (those along which we cannot send any more flow) and adjust the remaining capacities appropriately.

Let us try the above idea on this simple example.



Suppose we choose the $s-a-b-t$ path and send 1 unit of flow along this path. The 3 edges (s,a), (a,b), and (b,t) are saturated and so we delete them. The graph becomes:



There is no s-t path now & so the algorithm terminates. Our flow value is 1.

However the max flow value in our input graph is 2.

So our method of updating the graph is not correct. We need a method to correct our mistakes.

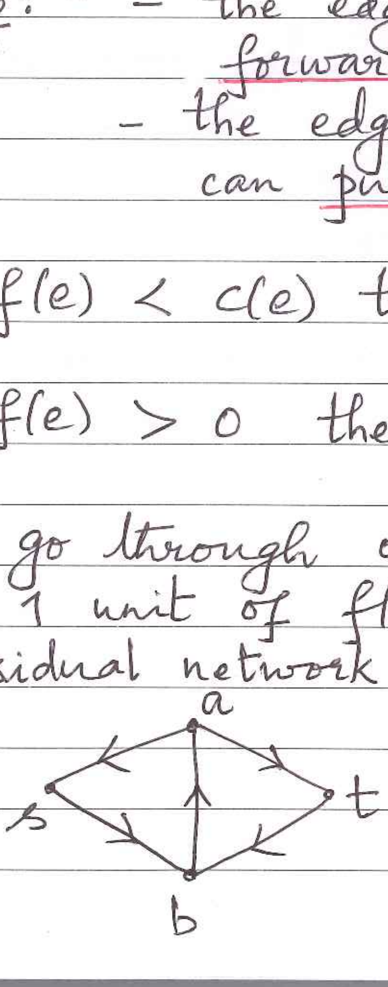
- for every edge (u, v) along which we send α units of flow (for any α), we should introduce a reverse edge (v, u) through which we can send back up to α units of flow. This allows us to backtrack and reverse the possibly wrong decisions that we made in the past.

Residual graph with respect to flow f
- this captures possible changes that can be made to flow f.

Call this residual graph or network G_f.
For every edge e in G, we have up to 2 edges in G_f:
- the edge e' denotes additional forward capacity
- the edge e'' denotes by how much we can pull back flow along e.

- If $f(e) < c(e)$ then e' has residual capacity $r(e') = c(e) - f(e)$
- If $f(e) > 0$ then e'' has residual capacity $r(e'') = f(e)$.

Let us go through our example again. After sending 1 unit of flow along the $s-a-b-t$ path, the residual network is the following graph:



- all residual capacities are 1.
- there is an s-t path here: $s — b — a — t$.
- so we can augment flow f.

Claim. If t is reachable from s in G_f then f is not a maximum flow.

Proof. We will augment f as follows:
- let p be any simple s-t path in G_f
- let δ = minimum residual capacity of any edge of p.
- construct a flow f' as follows:
  - $f'(e) = f(e) + \delta$ if $e' \in p$
  - $f'(e) = f(e) - \delta$ if $e'' \in p$
  - $f'(e) = f(e)$ if neither e' nor e'' is in p

Exercise. Show that f' is a valid flow in G. That is, show that (1) $0 \leq f'(e) \leq c(e) \ \forall e \in E$ and (2) flow conservation constraints are obeyed by all vertices, i.e. total flow entering u (other than s and t) = total flow leaving u for $u \in V - \{s, t\}$.

We will now show that value(f') = value$(f) + \delta$.

$$\text{value}(f') = \text{excess}(t) = -\text{excess}(s)$$
$$= \sum_{e:\,\text{start}(e)=s} f'(e) - \sum_{e:\,\text{end}(e)=s} f'(e)$$

Since p is a path in G_f from s to t, there is either some e' with start$(e) = s$ or some e'' with end$(e) = s$ in p.

- in the first case, i.e., when $e' \in p$,
$$\text{value}(f') = \sum_{e:\,\text{start}(e)=s} f(e) + \delta - \sum_{e:\,\text{end}(e)=s} f(e)$$
$$= \text{value}(f) + \delta.$$
- in the second case, i.e., when $e'' \in p$,
$$\text{value}(f') = \sum_{e:\,\text{start}(e)=s} f(e) - \Big(\sum_{e:\,\text{end}(e)=s} f(e) - \delta\Big) = \text{value}(f) + \delta.$$

Ford-Fulkerson algorithm

Date

```
1. Initialize $f(e) = 0 \ \forall e \in E$
2. Repeat
       - construct the residual network $G_f$
       - find a simple s-t path in $G_f$
       - send as much flow as possible
           along this s-t path
               (this step updates or augments f)
   until there is no s-t path in $G_f$
3. Return f
```

The above algorithm looks very similar to our "first attempt" which was wrong. However the above algorithm is correct and we will prove this now.

Theorem. The following statements are equivalent.

(1) f is a max flow in G.
(2) There is no s-t path in G_f.
(3) There is an s-t cut $(S, V-S)$ such that capacity(S) = value(f).

We are quite familiar with cuts now. We know that corresponding to any cut C, there is a partition of the vertex set V into two non-empty parts A and $V-A$.

Here we are interested in s-t cuts. An s-t cut corresponds to a partition $(A, V-A)$ such that $s \in A$ and $t \in V-A$

$$\text{Capacity of cut } C = \sum_{e \in C} c(e)$$

← sum of capacities of edges crossing this cut